Identification and Application of Invariant Critical Paths under NBTI Degradation

AUTHOR(S):

BIAN, Song; MORITA, Shumpei; SHINTANI, Michihiro; AWANO, Hiromitsu; HIROMOTO, Masayuki; SATO, Takashi

CITATION:

BIAN, Song ...[et al]. Identification and Application of Invariant Critical Paths under NBTI Degradation. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences 2017, E100.A(12): 2797-2806

ISSUE DATE:

2017-12

URL:

http://hdl.handle.net/2433/229140

PAPER *Special Section on VLSI Design and CAD Algorithms*

Identification and Application of Invariant Critical Paths under NBTI Degradation

Song BIAN[†a)], ***Nonmember***, **Shumpei MORITA**[†], ***Student Member***, **Michihiro SHINTANI**[†], **Hiromitsu AWANO**[†], **Masayuki HIROMOTO**[†], ***and*** **Takashi SATO**[†], ***Members***

SUMMARY As technology further scales semiconductor devices, aging-induced device degradation has become one of the major threats to device reliability. In addition, aging mechanisms like the negative bias temperature instability (NBTI) are known to be sensitive to workload (i.e., signal probability) that is hard to be assumed at design phase. In this work, we analyze the workload dependence of NBTI degradation using a processor, and propose a novel technique to estimate the worst-case paths. In our approach, we exploit the fact that the deterministic nature of circuit structure limits the amount of NBTI degradation on different paths, and propose a two-stage path extraction algorithm to identify the invariant critical paths (ICPs) in the processor. Utilizing these paths, we also propose an optimization technique for the replacement of internal node control logic that mitigates the NBTI degradation in the design. Through numerical experiment on two processor designs, we achieved nearly 300x reduction in the sheer number of paths on both designs. Utilizing the extracted ICPs, we achieved 96x–197x speedup without loss in mitigation gain.

key words: *NBTI, aging effect, invariant critical path, processor*

1. Introduction

The scaling of semiconductor devices is still continuing despite the fact that the devices manufactured become much less predictable. Statically, the uncontrollable shape of devices causes local variations that greatly alter their performance. Dynamically, materials age through a stochastic process, making the predictive calculation of the lifespan of devices much harder. While traditional methods rely on the concept of critical paths where a set of paths can be identified as being timing-critical, dynamic variations, also known as aging, degrade arbitrary paths on a per-chip level. Thus, it has become much harder to identify which path is the "critical paths" in the design.

Among various aging mechanisms, negative bias temperature instability (NBTI) is considered to be one of the most crucial factors that shorten the lifespan of VLSI circuits. NBTI is known to degrade the threshold voltage (V_{th}) of the pMOS transistor over its lifetime. However, the exact amount of degradation depends on the amount of time that the pMOS is turned ON, and is thus hard to determine at design time. The conventional methods to mitigate the NBTI degradation are by reducing its ON time. These methods include the internal node control (INC) technique [1], the input vector control (IVC) method [2], and aging-aware logic synthesis proposed in [3]. Obviously, NBTI-aware timing calculations are required for these methods to correctly predict the worst-case path delay in the design, such that mitigation techniques or re-synthesis can be applied to the most-critical path in the design. Unfortunately, in general, existing approaches for NBTI-aware timing analysis is extremely time-consuming due to the fact that one needs to consider all probabilistic combinations of input probabilities. Some works (e.g., [1]) avoid this situation by simplifying the per-gate NBTI degradation to be completely no degradation or full degradation in advance. Nonetheless, for a real-world general-purpose processor, this is not likely to be the case.

To pursue a methodology to systematically characterize the worst-case path delay in the presence of NBTI degradation, we first published the preliminary result of the idea of invariant critical path [4]. In the paper, we observed the fact that certain paths in a design are much more likely to become critical under NBTI, no matter what kind of workload is given to the design. Following this fact, we proposed a two-stage extraction technique for paths that are invariantly critical under NBTI degradation, thus named invariant critical path (ICP). By applying our proposed two-stage extraction algorithm and conducting experiment on two processors, we found that only a tiny portion of paths can be extracted from the large number of near-critical paths. While this is only an empirical observation, we argue that *practically*, ICPs are likely to be small in number. In other words, when the workload applied to the design changes drastically, the delay for each path may change significantly, but the relative path delay between paths stays still. By exploiting this property, we can identify the ICPs under NBTI degradation, regardless of the workload being applied to the design.

Upon identifying the ICPs, we integrate our extracted ICPs in the INC optimization process [5], [6]. While the aforementioned INC technique is a known technique to mitigate NBTI degradation, as discussed in Sect. 2.4, to find the optimal gates to be replaced by INC logic is a hard decision problem. Existing studies either base their optimization technique on unrealistic assumptions (e.g., in [5] as described), or only evaluate their technique using small circuits while relying heavily on circuit standby (e.g., in [7]). In addition, existing studies fail to take into account the fact that NBTI degradation can be highly dynamic, especially for a general-purpose processor without standby time. Through experiment, we show that using the extracted ICPs, INC op-

Manuscript received March 14, 2017.
Manuscript revised July 10, 2017.

[†]The authors are with Department of Communications and Computer Engineering, School of Informatics, Kyoto University, Kyoto-shi, 606-8501 Japan.
a) E-mail: paper@easter.kuee.kyoto-u.ac.jp

DOI: 10.1587/transfun.E100.A.2797

timization involving highly dynamic NBTI degradation can be easily solved by a simple exhaustive search algorithm on the identified ICPs. We actually achieved better mitigation gain across our test samples with better runtime compared to [6], which only considered one single primary-input vector. While this is only a single use-case of our ICP extraction algorithm, it fully demonstrates the amount of useful information the proposed ICPs can provide, especially for NBTI mitigation techniques.

The key contribution of this work is summarized as follows:

- An important observation of the relative invariance of critical paths under NBTI stress, even when different workloads are applied.
- A novel clustering-based ICP extraction algorithm that significantly reduces the number of critical path candidates in a design. Through experiment, we demonstrate that less than 100 ICPs can be extracted from the large amount of near-critical paths in processors. The extracted ICPs are proved effective in the INC optimization process.
- The application of our extracted ICPs to optimize INC replacement. We show that using ICPs, INC optimization can be done with near-optimal mitigation gain with significantly reduced runtime.

The rest of the paper is organized as follows. First, the NBTI model and our preliminary research are presented in Sect. 2. Second, the proposed two-stage path extraction algorithm along with the ICP-based INC optimization method is described in Sect. 3. Third, an ICP-based optimization technique for the INC logic replacement is presented as one of the applications of ICPs. Forth, the details of our experiment will be discussed in Sect. 5. Finally, the paper is concluded in Sect. 6.

2. Background and Motivation

In this section, we first explain the terms we use throughout the paper, and the NBTI model used for our experiment. We then discuss the preliminary research conducted that becomes the main motivation of this work.

2.1 Terminology

Through the paper, we use the term *invariant critical paths (ICPs)* to refer to this small subset of paths that are likely to become the most timing critical under NBTI degradation, after a certain time of use. It is noted, however, that the ICPs extracted in our analysis, in general, are not the most critical paths in a fresh chip without NBTI degradation. *Workload* here refers to the probabilities of the *primary inputs* of the processor, which are the abstraction of running different application programs with different input data, and is also occasionally referred to as the *primary-input vector*. *Signal probability* is mainly used to indicate the probability of a signal being in stress bias (logical zero for NBTI) for the connecting gates, and is annotated for each signal. The signal probabilities are determined by the primary-input probabilities through probability propagation, and the NBTI degradation for each gate along paths is then determined by the signal probabilities annotated on this gate.

2.2 NBTI Model

To predict NBTI-induced V_{th} degradation, NBTI measurements and mathematical models are studied in [8], [9]. Our degradation calculation is based on an analytical model in [8]. The NBTI-induced V_{th} degradation at a given signal probability α is shown in the following equation.

$$|\Delta V_{\text{th}}(\alpha)| \approx \left(\frac{0.001n^2K_v^2\alpha Ct}{0.81t_{\text{ox}}^2(1-\alpha)}\right)^n \quad (1)$$

where K_v is a function of gate-source voltage, V_{th}, and temperature. t_{ox} is the oxide thickness, n is the time exponent which holds the value 1/6, α expresses the signal probability of a pMOS transistor, and C is a function of temperature. When $\alpha = 100\%$, $|\Delta V_{\text{th}}|$ becomes infinite and the model becomes incorrect. In a similar manner to what has been shown in [10], an upper limit is thus defined by

$$|\Delta V_{\text{th}}| = (K_v^2 t)^n \quad (2)$$

2.3 Motivation

To obtain a more concrete idea on the distribution of signal probabilities in real-world designs, we conducted a preliminary research on an example five-stage pipelined processor [11]. The workload to this processor is application programs from MiBench [12] and GAUT [13]. In this experiment, we assumed that the processor will run a single program over a period of 10 years under 400 K temperature. For brevity, we leave out the details of the experiment here, and they can be referenced in Sect. 5.

Figures 1 and 2 are the distributions of the means and the standard deviations of all signal probabilities in the processor design, across various application programs. As marked in the figures, most signal probabilities are stable, i.e., their standard deviations are close to 0. Furthermore, the majority of the signals remain *static*, where their means close to either 0 or 1 with a 0 standard deviation. However, in terms of path delay, as Table 1 demonstrates, the worst-case path delay varies significantly from program to program. While the difference in path delay is expected, an important observation here is that the worst-path ID column shows some localities: under 8 different kinds of workload, only 4 distinct paths have became the worst-case path. This behavior hints on the fact that, although the worst-path delay varies greatly (from 5.8 ns to 6.8 ns), the worst-path ID is somewhat invariant. The invariance of path ID motivated us to explore the relationship between NBTI-induced path delay and workload, especially the relationship between primary-input probabilities and critical paths.

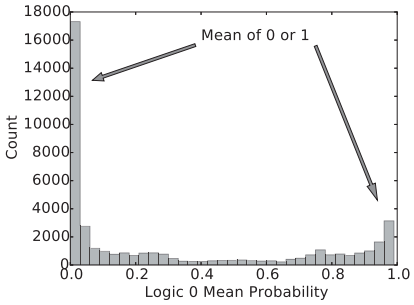

Fig. 1 Mean of signal probability obtained from application program simulation.

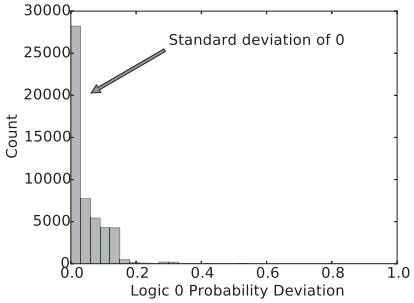

Fig. 2 Standard deviation of signal probability obtained from application program simulation.

Table 1 Worst-case path delay and worst-case path ID corresponding to different application programs

Program	Worst-path ID	Worst-path delay [ns]
lms	3042	5.8651
aes	2929	6.0152
sobel	2929	6.0152
qsort	3042	6.0158
cordic	417	6.0399
sieve	441	6.3299
fft	417	6.4929
conv3x3	2929	6.6932

In sections that follow, it is eventually discovered that, although path delays change significantly as primary-input probabilities change, the paths that degrade the most under NBTI is not changing. We define this behavior to be the *invariance* of critical paths under different workloads.

2.4 Internal Node Control

The general idea of the INC technique is to replace an existing logic gate that is upstream to the stressed gate with the INC logic, which is functionally equivalent to the original one but adding a recovery signal input to force its output value to be logical one [5]. With the mitigation signal, INC logic decreases the signal probabilities of downstream gates.

Figure 3 shows one implementation of INC logic. A pMOS transistor and an nMOS transistor are added to the original logic gate. The additional pMOS transistor connects output node to the supply rail in parallel to the pull up network (PUN) of the original logic gate. It is easy to see that as the

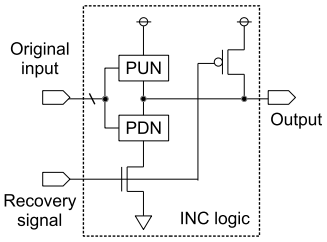

Fig. 3 General structure of the INC logic.

recover signal is asserted (logical "0"), the output of the INC logic is forced to be "1", and when the recovery signal is de-asserted (logical "1"), the INC gate functions as the original logic gate.

The INC method can effectively mitigate local NBTI degradation. However, one important challenge to the effective utilization of the method is that it is extremely hard to find the best gate to replace with INC. The amount of NBTI degradation varies on a per-gate scale, and even when the amount of degradation is fixed, the optimization problem is still known to be NP-complete [14]. Previous studies on the mitigation of NBTI using INC either focused on the so called "static NBTI" where only the standby-time NBTI is considered, and the amount of NBTI degradation is fixed to either no or full degradation [1], or assumed that the primary-input vector to the design converges to a single probability value [6]. For a general purpose processor, neither assumption holds; the processor can constantly operate without any standby time, and workloads change dramatically across different usage of the processor. By integrating our invariant-path extraction algorithm into the mitigation flow, however, this problem can be solved easily. The details of our proposed cluster-based INC optimization will be discussed in Sect. 4.

3. Invariable Critical Path Analysis

In this section, the proposed two-stage path extraction process is introduced. Figure 4 presents an overview of the proposed technique. In the beginning, dataset generation produces a set of data, where each piece of data refers to a set of signal probabilities for all signals in the target design. Each piece of data is generated in probability propagation, where one particular random primary-input vector is propagated through the design. A *dataset* then is defined as a set of such data. Supposedly, the data in the dataset represent various applications run on the processor and their inputs. After generating the dataset, each signal in the target design will obtain a set of gate probabilities. The timing analysis with NBTI step is then applied to the dataset to obtain a set of path delay distributions for the paths in the target design. Next, a two-stage path extraction is performed to extract the ICPs in the design. After the path extraction, error and path reduction analysis is performed to ensure that the extracted paths correctly capture the worst-case path delay generated in the dataset. When the error converges, the extracted paths

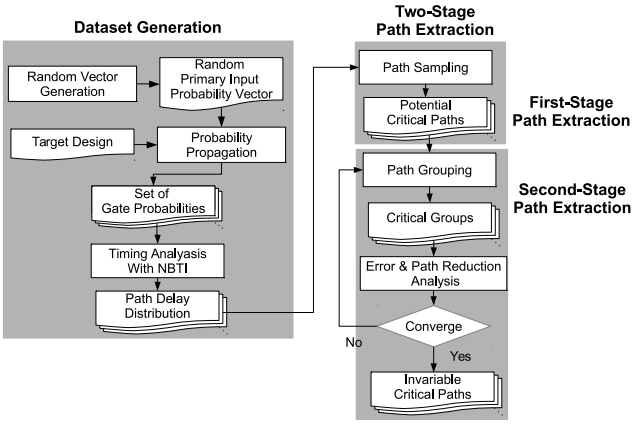

Fig. 4 Overview of the proposed technique, which includes a dataset generation process and a two-stage path extraction algorithm.

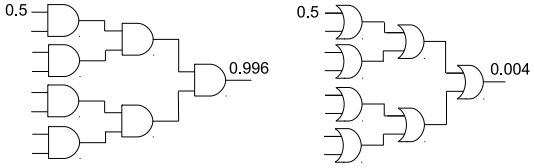

Fig. 5 An example of an AND tree and an OR tree where the inputs are set to have a probability of 0.5 of being at logic 0. It can be observed that the AND tree amplifies the probability of logic 0, and the OR tree decreases the probability of logic 0.

are then the extracted ICPs in this design.

3.1 Path Extractability

Before getting into the discussion of the path-extraction algorithm, an important assumption needs to be explained; that is, if the design is extractable. Here, an extractable design is defined as a design where a *small proportion* of its critical paths are *invariant* under workload-induced NBTI degradation. This definition coincides with the definition of ICP in Sect. 1. Thus, an extractable design always contains ICPs, and vice versa.

Under some special design decision, as NBTI degrades the circuit, all paths have nearly equal chances to become the most critical path. This design is not extractable. However, it is also *special* since this condition requires all paths to be virtually the same, such that the amount of degradation in a particular path solely depends on the workload being applied to it. NBTI degradation in such design is trivial to estimate in such case (e.g., clock tree, and multiple ALU units), where one path/unit can represent all other paths/units in the design. On the other hand, *general* designs tend to have distinct paths, and here, distinct paths are defined as paths that share very little gates in common. In such case, even if the paths have the same depth (i.e., same number of stages of gates), they may have extremely different NBTI degradation pattern, and is thus extractable.

We take Fig. 5 as an example for extractable design. Assuming that the AND and OR gates roughly have the same stage delay d, and that there is some function $\Delta d = f(p)$ that maps the probability value p to a delay shift Δd, the delay of each path in the AND and OR trees can be expressed as follows:

$$d_{\mathrm{AND}} = d + f(0.5) + d + f(0.75) + d + f(0.996) \quad (3)$$

$$d_{\mathrm{OR}} = d + f(0.5) + d + f(0.25) + d + f(0.006) \quad (4)$$

$$d_{\mathrm{diff}} = f(0.75) - f(0.25) + f(0.996) - f(0.006) \quad (5)$$

It is easy to see that the difference between these two delay values $d_{\mathrm{diff}} = d_{\mathrm{AND}} - d_{\mathrm{OR}}$ is always positive, so long as f is a monotonically increasing function, which is the case for NBTI. Thus, the idea behind our proposed method is to completely ignore the chance that any path in the OR tree can become the critical path, and we can extract only the paths in the AND tree for later timing analysis. This design containing an AND tree and an OR tree is thus considered an extractable design. From hereon in this paper, the target design is assumed to be extractable.

3.2 Dataset and Empirical Variables

Signal probability propagation has been studied extensively in the field of power analysis [15] and NBTI-related probability propagation [16]. In this paper, a similar approach is taken in calculating the propagation of signal probabilities from the primary inputs. The generated dataset then acts as a series of independent Monte-Carlo samples for determining parameters for the proposed path extraction algorithm. These samples are then analyzed using an NBTI-aware timing analysis tool. Through timing analysis, gate delays will be calculated, and if we add up the gate delays for each path, a collection of path delay distributions can be obtained. This set of path delay distributions, denoted as $\mathcal{D}_{\mathcal{P}_{\mathrm{all}}}$, works as a training dataset for the path extraction algorithm, and helps to determine the empirical variables T^{emp}. The threshold delay T^{emp} is used in the first-stage path extraction and will be explained in Sect. 3.3.1.

3.3 Extraction Algorithm

After calculating the timing distribution of the target design under different workloads using the previously mentioned timing analysis, path extraction can finally be performed. The two-stage path extraction algorithm is summarized in Algorithm 1. The algorithm can be better understood through Fig. 4. In Algorithm 1, lines 1–5 correspond to the first-stage path extraction, and lines 6–15 complete the second stage of the path extraction. Lines 16–22 are the error analysis step, and the algorithm repeats until error converges or $k = N$. The inputs to the algorithm include the predetermined T^{emp}, the delay distribution of all paths in the design $\mathcal{D}_{\mathcal{P}_{\mathrm{all}}}$, the path list that contains all paths in the design $\mathcal{P}_{\mathrm{all}}$, as described in the previous Sect. 3.2. An additional term ε is added for error analysis, which will be discussed in Sect. 3.3.2.

Algorithm 1 Two-stage path extraction

Require: T^{emp}, $\mathcal{D}_{\mathcal{P}_{\text{all}}}$, $\mathcal{P}_{\text{all}}$, ε
for each path $\in \mathcal{P}_{\text{all}}$ **do**
  for each $D \in \mathcal{D}_{\mathcal{P}_{\text{all}}}$ **do**
    if D(path) has value $> T^{\text{emp}}$ **then**
      add path to path_list
    end if
  end for
end for
initialize $k = 0$
while (max_error $< \varepsilon$) and ($k < N$) **do**
  increment k
  initialize Rep_List
  $\mathcal{G}_{\text{all}}$ = partition path_list into k groups
  for each $G \in \mathcal{G}_{\text{all}}$ **do**
    for each $D \in \mathcal{D}_{\mathcal{P}_{\text{all}}}$ **do**
      order paths in G by their delays in D
      increment the count for the most critical path in G according to D
    end for
    P_{REP} = path with the maximum count in G
    Rep_List.append(P_{REP})
  end for
  max_error $= -1$
  for each $D \in \mathcal{D}_{\mathcal{P}_{\text{all}}}$ **do**
    worst = Worst path delay in D
    worst_rep = Worst path delay in D only considering paths in Rep_List
    error = worst - worst_rep
    if error $>$ max_error **then**
      max_error = error
    end if
  end for
end while
Rep_List, k

3.3.1 First-Stage Path Extraction and T^{emp}

The value of T^{emp} is chosen based on $\mathcal{D}_{\mathcal{P}_{\text{all}}}$, the path timing distribution of the generated dataset as (a) in Fig. 6. At first, the choice of T^{emp} can be made arbitrarily. A simple path sampling is carried out once T^{emp} is determined. The main purpose of this stage is to reduce the large number of paths, such that later stages of the algorithm run faster and more efficient. The sampling process basically extracts paths that, according to the result of timing analysis, have ever crossed the barrier determined by T^{emp}. Larger T^{emp} reduces the number of paths to be considered significant, but increases the chance of missing paths that *could* have a chance to become the most critical path under some primary-input patterns that have not been applied (e.g., path 1 is left out with T_0^{emp}, but it could become the most critical path under some primary-input vector). However, it is also likely that path 1 can never obtain a longer path delay than path 2 and path 3, if the real circuit looks like the one in Fig. 6(b). Hence, by the assumption of extractable design, if the selected T^{emp} correctly captures all possible critical-path candidates (path 2 and path 3), further deduction in T^{emp} will result in no change in the result of the two-stage path extraction. Thus, by performing the proposed path extraction algorithm on several T^{emp} steps, the true empirical delay boundary $T^{\text{emp}}_{\text{true}}$ can be found and verified.

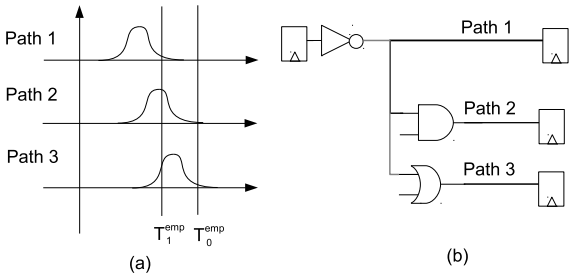

Fig. 6 An example of three paths with two different T^{emp} choices. (a) is an example of path delay distribution where the x-axis is delay. Depending on T^{emp}, path 1 will or will not be extracted in the first stage. (b) is an example circuit that could have a delay distribution in (a).

3.3.2 Second-Stage Path Extraction

Although the first stage of the proposed path-extraction algorithm already reduces the number of paths to be considered significant, it fails to take into account the correlation between paths. That is, suppose two paths P_a and P_b can both have their delays to be greater than T^{emp}, both of them are assumed to be paths that have a chance to become the most critical paths in the design. However, if P_a and P_b are mostly the same, with only a slight difference at some stages along the path, the delays of these two paths are going to be extremely close. In addition, if, in stages when P_a and P_b have different gates and the delays for the different gates in P_a are always greater than P_b (e.g., AND gate versus inverter), then P_b essentially has no chance to become the most critical path in the design, regardless of the primary-input pattern (i.e., workload). Thus, to further reduce the number of paths needed to be considered, the second stage of the path-extraction algorithm is performed.

The main procedure in the second-stage path extraction is to partition the paths selected in the first stage into groups, and select a representative path for each group. This grouping process can be modeled as a classic multi-class classification problem and potentially be solved by some advanced machine learning algorithm. In this paper, we used the simple yet effective k-means++ clustering algorithm for path partitioning [17]. The k-means++ algorithm is a popular algorithm mainly used in data mining [17] that partitions n observations into k sets, where each observation is a d-dimensional vector. The path vector $\mathbf{p}$ used in this study is an N-dimensional vector consisting of path similarity measured in the Jaccard similarity coefficient [18].

$$\mathbf{p_i} = (J_{i0}, J_{i1}, \cdots, J_{iN}), \text{ where} \tag{6}$$

$$J_{ij} = \frac{G_i \cap G_j}{G_i \cup G_j} \tag{7}$$

In (6), N is the number of paths after the first-stage of path extraction algorithm, J_{ij} represents the Jaccard similarity coefficient between path i and path j. J_{ij} can be calculated

from (7), and G_i in the equation represents the set of gates contained in path i. Thus, each similarity index J_{ij} is measured by the number of gates shared (intersection) by path i and j, and the total number of distinct gates (union) in path i and j.

Since the k-means++ algorithm works only as a classifier, it is necessary to elect a "real" representative path from each group, instead of the centroid point. This representative election actually determines the maximum error. For example, if P_b in the previous example is selected as a representative instead of P_a, the delays of this path group is going to be constantly underestimated. In this work, we used a simple majority approach for representative election, as lines 11–15 indicate in Algorithm 1. The result of timing analysis is again used, and the path in the group that has the highest number of times being the worst-delay path in the group is selected to be the representative.

After the group representatives are elected, the maximum error is calculated in lines 16–22 in Algorithm 1, where the difference between the true worst-case path delay of all paths in one piece of data D is compared to the worst-case path delay of the paths in the representative list. This error needs to be small, and a constraint ε is added to make sure that the maximum error is below this value. If the error is larger than ε, we consider the number of groups for the k-means++ algorithm to be insufficient, such increment the number of groups by one and repeat the grouping process. Due to the stochastic nature of the dataset generation, for extremely small number of datasets, this approach may result in more groups being used than necessary. However, with a reasonable amount of datasets that correctly captures the relative criticalness of each path, as will be described in Sect. 5.3, correct representative-path selection results in a quicker convergence of training (as well as testing) error.

4. Cluster-Based INC Optimization

In this section, we give a simple overview of the proposed INC optimization method based on ICP. The method is described in Algorithm 2. It is a simple algorithm that basically contains a nested loop to generate a list of M gates to be replaced by INC logic. The algorithm requires two inputs, the Rep_List generated by Algorithm 1, and M. M here is the number of maximum number of gates permitted to be replaced by INC, and can be determined as a design trade-off as in the traditional INC optimization techniques [5], [6]. Basically, more INC means better NBTI mitigation, but also more power consumption. Moreover, when the worst-case path delay cannot be further reduced, adding more INC becomes meaningless. This fact can serve as an upper bound for M.

As for the algorithm, we first consider all gates in the extracted ICPs to be replacement candidate for the INC logic. In Algorithm 2, the outer loop, which starts from line 3, simply repeats the inner loop of optimizing a single INC replacement iteratively from $i = 0$ to $M - 1$. The inner loop, described in lines 5–13, tries to find the best gate to be replaced by INC. The algorithm takes an exhaustive approach, where it replaces one gate with INC logic at a time, and recalculates the worst-case path delay after the replacement. In each round of the inner loop, a worst-case path delay, which corresponds to the INC replacement of a particular gate, is calculated. This process is repeated for all candidate gates, and the gate replacement that achieves the minimum worst-case path delay is selected to be replaced by INC and deleted from the candidate list. This process goes on until $i = M - 1$. The algorithm per se is trivial. Nevertheless, without ICP, we have to recalculate all path delays in the whole design, which is obviously impractical. With the small number of paths extracted as shown in Sect. 5, the algorithm demonstrates excellent performance in terms of both mitigation gain and runtime.

Algorithm 2 Cluster-based INC optimization

```
Require: ICP_List, M
 1: candidates = extract gate from paths in ICP_List
 2: i = 0
 3: while i < M do
 4:     Initialize d_min
 5:     for each g ∈ candidates do
 6:         replace g with INC
 7:         D = recalculate all path delays in ICP_List
 8:         d_worst = max(D)
 9:         if d_worst < d_min then
10:             d_min = d_worst
11:             INC_gate = g
12:         end if
13:     end for
14:     Remove g from candidates
15:     INC_List.append(INC_gate)
16:     i = i + 1
17: end while
18: return INC_List
```

It is important to note here that in the process of executing Algorithm 2, it is required to calculate the aged delays of gates in the design, which depends on the particular setting of probabilities on the primary inputs. Thus, an optimized solution on one particular setting may not be as optimized on another one. However, as we will show in Sect. 5.4, our empirical experiments indicate such approach suffices.

5. Experiment

In this section, we present the result of the invariant critical path extraction algorithm and the INC optimization algorithm. We first give a detailed result of the ICP extraction for one of our test designs in Sect. 5.3, and then illustrate the efficiency of our extracted ICP in INC optimization described in Sect. 5.4.

5.1 Experiment Setup

Numerical experiments are conducted using two example processors. One is a five-stage pipelined processor from a commercial IP library [19] (named "Shino"), and the other is a modified version of the five-stage pipelined processor

[11] that implements the full MIPS32 instruction set with a co-processor handling exceptions (named "Kotori"). In the experiment, Nangate 45 nm Open Cell Library [20] is used for circuit synthesis using a synthesis tool [21]. Gate-level probability is calculated using [22], and post-synthesis simulation is conducted using [23]. A static timing analysis (STA) tool [24] is used to extract paths from the processor. We have extracted the top 25,446 and 24,978 paths, ordered by worst-case slack, from Shino and Kotori, respectively. As later explained in Sect. 5.3, further increasing the number of paths extracted by the STA tool becomes irrelevant. For timing analysis with NBTI, a fast method described in [25] is implemented using the Python language, where the traditional STA library is replaced with a three-dimensional look-up table (LUT) that returns the gate delay with NBTI degradation when the input signal probabilities are given. The degradation condition, as also described in Sect. 2, is 400 K for 10 years. All experiments were conducted on Linux PC with Intel Xeon E5-2630 v2 2.60 GHz CPU, using a single thread. The ICP extraction and ICP-based INC optimization were implemented using the Python language.

5.2 Dataset Generation

To carefully evaluate the quality of the extracted invariant critical paths, we used a training-test approach, where a training set is used for path extraction, and a test set is used for invariant critical path evaluation. In this experiment, to each processor design, a training dataset of size 2,500 samples is generated, and the proposed two-stage path extraction process is applied. The extracted invariant critical paths are then evaluated on both the training set and a test dataset of size 7,500 samples (total of 10,000 samples). We also show that the result of random input vector applies to real-world application programs as well. The error bound ε is taken to be 1% (much smaller than the typical STA error).

Figures 7 and 8 are the results of timing analysis on the set of training + test (10,000) and training only (2,500) samples where only the delay of the worst-slack path is shown. The two figures share a similar shape of distribution. However, the largest delay recorded is drastically different. The largest delay observed in training dataset is 6.7 ns, while it is 9.0 ns for the test dataset. Comparing to the fresh-time worst-case delay (delay without NBTI degradation) of 4.7 ns, depending on the workload variation, the delay can be as much as 1.9x times longer than the fresh-time delay. Thus, the training dataset we used cannot capture the full degraded delay range. Nevertheless, due to the relative order property described in Sect. 1, by calculating the path delays of the selected paths, it is demonstrated in Sect. 5.3 that this value can be captured without considering a large amount of paths in the design.

5.3 Experiment Result: Path Extraction

We summarize the path extraction result in Table 2, where both designs tested achieved a significant amount of reduction in the number of critical paths. In what follows, we will discuss the detailed experiment result for Kotori, and since Shino draws extremely similar results, a detailed analysis is omitted.

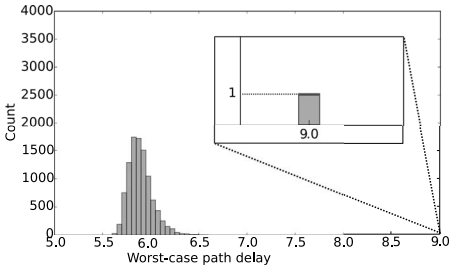

Fig. 7 Delay distribution of the training + test dataset (10,000 samples).

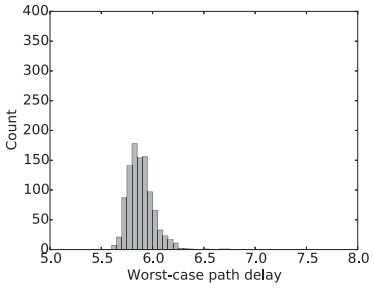

Fig. 8 Delay distribution of the training dataset (2,500 samples).

Table 2 Summary of path extraction result.

Design	Original paths	Extracted paths	Reduction
Shino	25,446	90	282x
Kotori	24,978	85	293x

Table 3 T^{emp} selection and the corresponding number of paths extracted.

T^{emp}		Path	Path	Converge	% Error
	Delay [ns]	1st stg.	2nd stg.		
$\mu + 3\sigma$	6.321	98	98	No	6.77
$\mu + 2\sigma$	6.180	142	142	No	6.15
$\mu + \sigma$	6.039	225	85	Yes	0.07
μ	5.898	405	85	Yes	0.07
$\mu - \sigma$	5.757	595	85	Yes	0.07
$\mu - 2\sigma$	5.616	890	85	Yes	0.07
$\mu - 3\sigma$	5.475	1510	85	Yes	0.07

To determine the T^{emp}, we assumed a normal distribution on the worst-case delay on the small dataset, and set T^{emp} according to the μ and σ of the normal distribution. It is noted that the worst-case delay does not necessarily obey a normal distribution, and the T^{emp} selection can be made using any approach. Our T^{emp} choices, their corresponding delays and the number extracted in the first stage as well as in the second stage are summarized in Table 3. It can be observed that the number of paths extracted in the first stage increases in a log-like manner as we decrease the boundary delay. However, as the result from the second stage shows, only an extremely limited number of paths are actually important. Other paths are generally considered to be paths that can obtain a large path delay, but always less than the

delay of the invariant critical path. On the other hand, if T^{emp} is too large, as in the case of $\mu + 3\sigma$ and $\mu + 2\sigma$, the error never converges, and the path extraction algorithm actually fails at these T^{emp} choices. Overall, it can be observed that the training phase of the invariant critical path selection correctly captured the relative path orders between paths, resulting in negligible errors when evaluated on the full dataset (10,000 samples).

Figure 9 gives the result of second-stage path extraction for a single T^{emp} choice. The horizontal axis denotes the number of groups initially given for k-means++ classification, and the vertical axis shows delay error. The figure reveals that when distinctive paths are not separated, both training set and test set retain large errors. Once the necessary paths are separated, the errors drop significantly below ε, which is 1%. The same result is obtained evaluating application programs.

To echo with the preliminary research in Sect. 2, the extracted 85 paths are evaluated against all paths using different signal probabilities determined by each application programs. Table 4 summarizes the result for each application. It is clear from the table that worst-path delays are completely captured by the extracted invariant critical path with a 1% ε (actually ε can be as small as 0.1%, since all errors are around 0.08% at 85 groups). Note that these errors are all results evaluated on the full dataset with 10,000 samples.

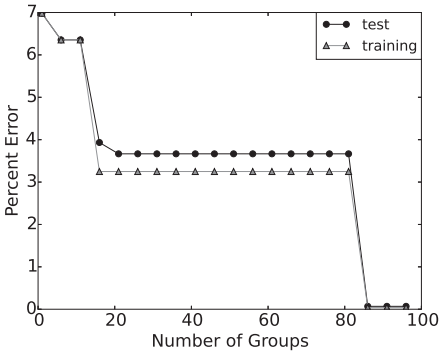

Fig. 9 Maximum percent error calculated using training and test datasets as the number of groups used for k-means++ algorithm increases.

Table 4 Summary on the worst-case delay evaluated on the extracted invariant critical paths for different programs.

Program	Extracted worst ID	Extracted delay [ns]	True delay [ns]	% Error
lms	2929	5.8602	5.8651	0.08
aes	2929	6.0152	6.0152	0
sobel	2929	6.0152	6.0152	0
qsort	2929	6.0152	6.0158	0.008
cordic	417	6.0399	6.0399	0
sieve	417	6.3255	6.3299	0.07
fft	417	6.4929	6.4929	0
conv3x3	2929	6.6932	6.6932	0

5.4 Experiment Result: INC Optimization

Figure 10 depicts the NBTI mitigation effect of INC logic over the 10,000 random samples generated for Shino. A significant amount of NBTI mitigation is observed in all samples. To further demonstrate that the extracted ICPs can fully represent the design even when INC presents, we adjusted line 7 in Algorithm 2 to recalculate the delays for all paths in the whole design, instead of only in ICP_List. The resulting algorithm (temporarily named the AP-based INC optimization) is a trivial brute-force optimization method that is extremely time consuming to perform.

Table 5.3 gives a summary for the result of INC optimization, and Table 6 provides information on the runtime of dataset generation and two-stage path extraction. The number of gates that are replaced with INC is drastically different across design, which is expected. With similar post-INC mitigation gain, the runtime reduction, without dataset generation time, for the ICP-based INC optimization is 197x for Shino, proving the efficiency of the ICP-based method. Since we consider ICP to be available at the time of INC optimization, runtime for dataset generation and two-stage path extraction is not included in the runtime reduction. However, as Table 6 indicates, even including the ICP-extraction time, we can still achieve 74x runtime reduction on Shino, for ex-

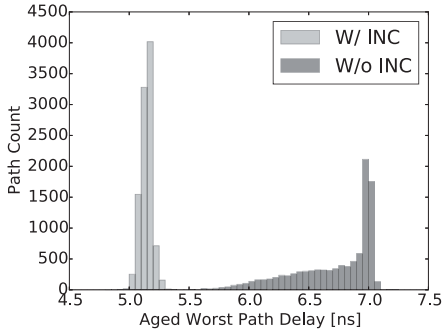

Fig. 10 Worst-case path delay of Shino evaluated on all samples (training + test) before and after INC mitigation optimized on ICPs.

Table 6 Summary of dataset generation and path extraction runtime result

Design	Dataset Generation	Path Extraction
Kotori	4744 s	4619 s
Shino	2029 s	6227 s

Table 5 Summary of INC optimization result

Design	# INC	Worst delay w/o INC	Worst delay ICP-based INC	Worst delay AP-based INC	ICP Runtime	AP Runtime	Runtime reduction
Kotori	6	9.02 ns	6.24 ns	6.14 ns	274.7 s	26513 s	96x
Shino	50	7.19 ns	5.55 ns	5.57 ns	3104.4 s	612874 s	197x

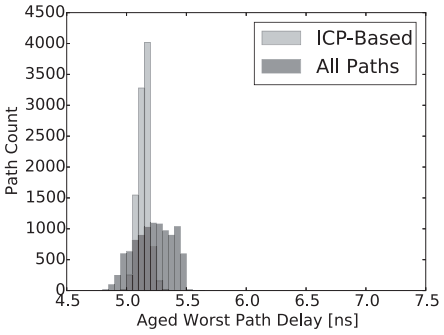

Fig. 11 Worst-case path delay of Shino evaluated on all samples (training + test) after INC mitigation.

ample. This is because the runtime of such generation is only related to circuit size, and not how hard it is to optimize INC in the design. In the case of Shino, an ICP extraction is clearly much more efficient than an all-path-based method.

It is observed that, in the case of Shino, the ICP-based INC replacement achieves a better mitigation gain compared to AP-based method. This is due to the fact that while the INC optimization is conducted using only one probability sample as described in Sect. 4, the results in Table 5.3 is the worst-case result evaluated across 10,000 samples. In addition, as Fig. 11 illustrates, in terms of mitigation gain, ICP-based method actually performs equally good or better in the majority of cases for Shino.

6. Conclusion

In this paper, we proposed a novel approach for the extraction of ICPs under NBTI degradation, and applied the extracted ICPs to the INC optimization problem. Motivated by the fact that the number of invariant critical paths may be extremely small, we utilized a systematical approach of dataset generation and two-stage path extraction to carefully classify and extract the invariant critical paths from a design, ensuring these extracted paths really represent the whole group. By conducting a numerical experiment, we extracted the ICPs in two example five-stage pipelined processors, and verified the extracted paths on a test dataset. We have shown that with a tiny error bound ($< 1\%$), worst-case path delay can be successfully obtained by only calculating the delays of less than 100 paths in a large processor design. Finally, to truly demonstrate the invariability of our ICPs, we use the ICPs to optimize the INC replacement, and achieved the same level of mitigation gain with over 96–197x speedup.

Acknowledgments

This work was partially supported by JSPS KAKENHI Grant No. 26280014, 17H01713, 15K15960, and Grant-in-aid for JSPS Fellow (DC1). The authors also acknowledge support from VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Synopsys, Inc.

References

[1] D.R. Bild, R.P. Dick, and G.E. Bok, "Static NBTI reduction using internal node control," ACM Transactions on Design Automation of Electronic Systems (TODAES), vol.17, no.4, p.45, 2012.
[2] Y. Wang, H. Luo, K. He, R. Luo, H. Yang, and Y. Xie, "Temperature-aware NBTI modeling and the impact of input vector control on performance degradation," Proc. IEEE Design Automation and Test in Europe, pp.1–6, 2007.
[3] M. Ebrahimi, F. Oboril, S. Kiamehr, and M.B. Tahoori, "Aging-aware logic synthesis," Proc. IEEE/ACM International Conference on Computer-Aided Design, pp.61–68, 2013.
[4] S. Bian, M. Shintani, S. Morita, M. Hiromoto, and T. Sato, "Workload-aware worst path analysis of processor-scale NBTI degradation," Proc. Great Lakes Symposium on VLSI, pp.203–208, 2016.
[5] D.R. Bild, R.P. Dick, and G.E. Bok, "Static NBTI reduction using internal node control," ACM Trans. Design Automation of Electronic Systems, vol.17, no.4, pp.45–75, 2012.
[6] S. Bian, M. Shintani, Z. Wang, M. Hiromoto, A. Chattopadhyay, and T. Sato, "Mitigation of NBTI-induced timing degradation in processor," Proc. IEEE Asian Test Symposium, pp.234–239, 2016.
[7] Y. Wang, X. Chen, W. Wang, Y. Cao, Y. Xie, and H. Yang, "Leakage power and circuit aging cooptimization by gate replacement techniques," IEEE Trans. Very Lagre Scale Integr. (VLSI) Syst., vol.19, no.4, pp.615–628, 2011.
[8] S. Bhardwaj, W. Wang, R. Vattikonda, Y. Cao, and S. Vrudhula, "Predictive modeling of the NBTI effect for reliable design," Proc. IEEE Custom Integrated Circuits Conference, pp.189–192, 2006.
[9] H. Awano, M. Hiromoto, and T. Sato, "BTIarray: A time-overlapping transistor array for efficient statistical characterization of bias temperature instability," IEEE Trans. Device Matel. Rel., no.3, pp.833–843, 2014.
[10] H. Konoura, Y. Mitsuyama, M. Hashimoto, and T. Onoye, "Comparative study on delay degrading estimation due to NBTI with circuit/instance/transistor-level stress probability consideration," Proc. IEEE International Symposium on Quality Electronic Design, pp.646–651, 2010.
[11] "OpenCores," http://www.opencores.org
[12] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, "MiBench: A free, commercially representative embedded benchmark suite," Proc. IEEE International Workshop on Workload Characterization, pp.3–14, 2001.
[13] C. Chavet, C. Andriamisaina, P. Coussy, E. Casseau, E. Juin, P. Urard, and E. Martin, "A design flow dedicated to multi-mode architectures for DSP applications," Proc. IEEE/ACM International Conference on Computer-Aided Design, pp.604–611, 2007.
[14] D.R. Bild, G.E. Bok, and R.P. Dick, "Minimization of NBTI performance degradation using internal node control," Proc. IEEE Design Automation and Test in Europe, pp.148–153, European Design and Automation Association, 2009.
[15] F.N. Najm, "A survey of power estimation techniques in VLSI circuits," IEEE Trans. Very Lagre Scale Integr. (VLSI) Syst., vol.2, pp.446–455, 1994.
[16] A. Stempkovsky, A. Glebov, and S. Gavrilov, "Calculation of stress probability for NBTI-aware timing analysis," Proc. IEEE International Symposium on Quality Electronic Design, pp.714–718, 2009.
[17] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," Proc. ACM-SIAM Symp. Discrete Algorithms, pp.1027–1035, 2007.
[18] L. Hamers, Y. Hemeryck, G. Herweyers, M. Janssen, H. Keters, R. Rousseau, and A. Vanhoutte, "Similarity measures in scientometric research: the Jaccard index versus Salton's cosine formula," Information Processing Management, vol.25, no.3, pp.315–318, 1989.
[19] Synopsys, Inc., Processor Designer User Guide Version G-2012.09.
[20] Si2.org, "Nangate 45nm open cell library." http://www.si2.org
[21] Synopsys, Inc., Design Compiler User Guide Version I-2013.06.

[22] Synopsys, Inc., PrimeTime PX User Guide Version H-2013.06.
[23] Mentor Graphics, Inc, ModelSim SE 10.2c.
[24] Synopsys, Inc., PrimeTime Fundamental User Guide Version H-2013.06.
[25] S. Bian, M. Shintani, S. Morita, M. Hiromoto, and T. Sato, "Nonlinear delay-table approach for full-chip NBTI degradation prediction," Proc. IEEE International Symposium on Quality Electronic Design, pp.307–312, 2016.

Song Bian received B.S. from University of Wisconsin-Madison, Wisconsin, USA in 2014. He attended Kyoto University in 2015 in pursuing a master's degree in the field of computer engineering. His main areas of interest include electric design automation (EDA), processor architecture and hardware security. He is a student member of IEEE and IPSJ.

Shumpei Morita received B.E. degree in Electrical and Electronic Engineering from Kyoto University in 2016. He is a master course student at Department of Communications and Computer Engineering, Kyoto University.

Michihiro Shintani received B.E. and M.E. degrees from Hiroshima City University, Hiroshima, Japan, and a Ph.D. degree from Kyoto University, Kyoto, Japan, in 2003, 2005 and 2014, respectively. He was with Panasonic Corporation, Osaka, Japan, from 2005 to 2014, with Semiconductor Technology Academic Research Center (STARC), Yokohama, Japan, from 2008 to 2010, and with Kyoto University, Kyoto, Japan. In 2017, he joined the Graduate School of Information Science, Nara Institute of Science and Technolgy (NAIST), where he is currently an assistant professor. His research interests include reliability-aware LSI design and circuit simulation of power converters. He is a member of IEEE and IEICE. He received the IEEE Workshop on RTL and High Level Testing 2004 Best Paper Award, IEICE VLD Excellent Student Author Award for ASP-DAC 2014 and IEEE Kansai Section Student Paper Award 2014.

Hiromitsu Awano received his B.E. degree in Informatics and his master degree in Communications and Computer Engineering from Kyoto University in 2010 and 2012, respectively. Presently, he is a doctor course student at Department of Communications and Computer Engineering, Kyoto University. He is a research fellow of japan society for the promotion of science and a student member of IPSJ.

Masayuki Hiromoto received B.E. degree in Electrical and Electronic Engineering and M.Sc. and Ph.D. degrees in Communications and Computer Engineering from Kyoto University in 2006, 2007, and 2009 respectively. He was a JSPS research fellow from 2009 to 2010, and with Panasonic Corp. from 2010 to 2013. In 2013, he joined the Graduate School of Informatics, Kyoto University, where he is currently a senior lecturer. His research interests include VLSI design methodology, image processing, and pattern recognition. He is a member of IEICE and IPSJ.

Takashi Sato received B.E. and M.E. degrees from Waseda University, Tokyo, Japan, and a Ph.D. degree from Kyoto University, Kyoto, Japan. He was with Hitachi, Ltd., Tokyo, Japan, from 1991 to 2003, with Renesas Technology Corp., Tokyo, Japan, from 2003 to 2006, and with the Tokyo Institute of Technology, Yokohama, Japan. In 2009, he joined the Graduate School of Informatics, Kyoto University, Kyoto, Japan, where he is currently a professor. He was a visiting industrial fellow at the University of California, Berkeley, from 1998 to 1999. His research interests include CAD for nanometer-scale LSI design, fabrication-aware design methodology, and performance optimization for variation tolerance. Dr. Sato is a member of the IEEE and the Institute of Electronics, Information and Communication Engineers (IEICE). He received the Beatrice Winner Award at ISSCC 2000 and the Best Paper Award at ISQED 2003.